Discrete Vector Quantization for Arbitrary Distance Function Estimation

John Oommen
Sch. of Comp. Sci.
Carleton University
Ottawa, CANADA
oommen@scs.carleton.ca

İ. Kuban Altınel
Dept. of Indus. Eng.
Boğaziçi University
İstanbul, TÜRKİYE
altinel@boun.edu.tr

Necati Aras
Türk Elektrik End. A. Ş.
Topkapı 34020
İstanbul, TÜRKİYE
aras@boun.edu.tr

Abstract— *There are currently many vastly different areas of research involving adaptive learning. Two of these are the ones which concern neural networks and learning automata. This paper develops a method by which the general philosophies of Vector Quantization (VQ) and discretized automata learning can be incorporated for the computation of arbitrary distance functions. The latter is a problem which has important applications in Logistics and Location Analysis. The input to our problem is the set of coordinates of a large number of nodes whose inter-node arbitrary "distances" have to be estimated. To render the problem interesting, non-trivial and realistic, we assume that the explicit form of this distance function is both unknown and uncomputable. Unlike traditional Operations Research methods, which use optimized parametric functional estimators, we have utilized discretized VQ principles to first adaptively polarize the nodes into sub-regions. Subsequently, the parameters characterizing the sub-regions are learnt by using a variety of methods (including, for academic purposes a VQ strategy in the meta-domain). After an initial training phase, a system which achieves distance estimation attempts to yield an estimate of any node-pair distance without actually deriving an explicit form for the unknown function. The algorithms have been rigorously tested for the actual road-travel distances involving cities in Türkiye and the results obtained are conclusive. Indeed, these present results are the best currently available from any single or hybrid strategy.*

Keywords— *Artificial Intelligence, Location, Neural Networks, Discretized algorithms, Road Transportation, Self Organizing Maps, Vector Quantization.*

1 INTRODUCTION

There are currently many vastly different areas of research involving adaptive learning. Two of these are the ones which concern neural networks and learning automata. The Kohonen Network which uses the principles of vector quantization [25] has been proposed as a fundamental model for neural computing. It has also been used extensively in hundreds of applications. As opposed to this, the field of learning automata has demonstrated the power of working in a discretized space [43, 44, 45, 47] when interacting with a random environment. In this paper we develop a method by which the general philosophies of these families can be incorporated so as to yield enhanced algorithms for a problem which has received much attention in Logistics and Location Analysis namely, that of evaluating arbitrary distance functions [17, 33].

Artificial Neural Networks (NNs) are biologically inspired structures developed to mimic the functionality of the human brain. Even though the biological nature of the human brain is not thoroughly understood, researchers have developed these NNs which perform adequately in many application domains without many of the actual brain structures. Instead, most researchers developed structures to perform useful operations (brain-like functions i.e., the ability to learn from experience) without the cumbersome work of actually modelling the real network that exists in the human brain.

One of the most popular NNs is the Self-Organizing Map (SOM) popularized by Kohonen which has been used in a variety of applications. In statistical pattern recognition it has been used in the recognition of Finnish and Japanese speech [23, 26, 27], sentence understanding [55], in classification of sea-ice [49] and even in the classification of insect courtship songs [42]. From an hardware point of view the SOM has been used in the design of algorithms, which at the lowest level can control the production of semiconductor substrates [37, 57], and at a higher level the synthesis of digital systems [22]. It has also been used in solving certain optimization problems such as the Travelling Salesman Problem [51].

The beauty of the SOM is the fact that the individual neurons adaptively tend to learn the properties of the underlying distribution of the space in which they operate. Additionally, they also tend to learn their places topologically. This feature is particularly important for problems which involve two and three-dimensional **physical** spaces, and is indeed, the principal motivation for the SOM being used in path planning and obstacle avoidance in Robotics [19, 20, 38, 52, 53, 54].

As opposed to the families of NN, the study of the families of Learning Automata (LA) was developed by Tsetlin [58]. His intention was to model biological learning using a stochastic finite state machine interacting with a random environment. The LA selects an action from a finite set of possible actions. Feedback from the environment tells the LA if the chosen action was rewarded or penalized. The LA uses this information to decide which action to take next, and the cycle repeats itself. Learning automata and their applications have been reviewed by Lakshmivarahan [28], and by Narendra and Thathachar [41]. Learning automata are useful whenever complete knowledge about a stochastic environment is unknown, expensive to obtain or impossible to quantify. Thus they have found applications in various fields including game playing [41], pattern recognition [41], and object partitioning [48]. Learning automata are also useful when the characteristics of the environment with which they interact change during operation, and are thus useful in priority assignments in a queuing system [41], and the routing of telephone calls [41].

The aim of this paper is to attempt to develop a method by which the general philosophies of the SOM (or more precisely, the principles of Vector Quantization (VQ) as adapted by Kohonen in the SOM) and discretized automata learning can be utilized for fast distance function estimation.

We shall first formalize the problem being studied. Consider the situation in which a user is given a set of N nodes (cities), G, located in a multi-dimensional "physical" space. We assume that there is an unknown arbitrary distance function Φ between the nodes. By arbitrary, we mean that the set of inter-node distances dictated by Φ may or may not satisfy all the rigorous properties of a well-defined mathematical norm. Furthermore, the triangular inequality may also be violated. However, to keep the informal concepts of a distance measure valid, we impose the requirement that Φ is loosely related to the Euclidean norm as follows. First of all, $\Phi(P_i, P_i)$ must be zero, and $\Phi(P_i, P_j)$ must be symmetric. Furthermore, let P_i, P_j, P_m and P_n be any four nodes in G. Then, informally speaking, if the pairs (P_i, P_j) and (P_m, P_n) are "close" to each other in the physical world, the respective arbitrary distances between (P_i, P_m) and (P_j, P_n) must be correspondingly of similar magnitude. We formalize these concepts below.

Definition : A function Φ is defined to be a valid arbitrary distance function if for every P_i, P_j, P_m and P_n in S, the following is satisfied :

1. $\Phi(P_i, P_i) = 0$,
2. $\Phi(P_i, P_j) = \Phi(P_j, P_i)$, and,
3. For every $\epsilon > 0$ there exists a $\delta > 0$ such that
   $\| P_i - P_j \| < \epsilon$ and $\| P_m - P_n \| < \epsilon \Rightarrow |\Phi(P_i, P_m) - \Phi(P_j, P_n)| < \delta$.

In two earlier works [5, 46] we had demonstrated that the principles of VQ could naturally and powerfully be utilized to solve the arbitrary distance estimation problem. Indeed, the solution proposed in [5, 46] was a sequence of pattern recognition and polarizing modules governed by the laws of VQ. The salient contribution of this present paper is that we have shown that by merging the learning principles of two *families* of adaptive algorithms we can achieve an enhanced superior learning algorithm. Indeed, this is done by having the VQ operate in a discretized space, as wil be clarified presently. We shall refer to the new strategy as Discretized Vector Quantization (DVQ).

From a "naive" perspective it would appear that since we are working with a "real-life" physical world, the SOM would constitute a natural tool to achieve complete learning, classification and estimation. While this is, of course, true from a philosophic point of view, the fact that the arbitrary function

Φ is not explicitly related to their geographical (Euclidean) "as the crow flies" distance complicates the problem. Indeed, our earlier work in [5] demonstrated that an all-neural approach ([24] pp.82) is sometimes recommendable (as opposed to the speech recognition example discussed in [24]). But in our appplication domain, the results of [5] also clearly validated the hypothesis of Kohonen that a neural network be followed by a traditional strategy, because a neural preprocessor followed by a traditional optimization yielded even more superior results. In this paper we propose to pursue the point further - we shall show that if we can selectively take advantage of the principles of other learning paradigms, we can indeed guarantee an *even better* performance.

As in [5], the physical application domain in which we have tested our algorithms involves the actual road distances between the major towns in Türkiye. This has provided us with a platform to verify the power of our algorithms, and also to compare them to the results obtained using the existing techniques. [1] We are currently working on estimating the monetary cost (as the arbitrary "distance" function) of road travel in Türkiye and in the estimation of inter-string likelihood functions using analogous algorithms.

In all brevity we shall list the salient contributions of the paper. To the best of our knowledge, our strategy is the first reported technique which tackles distance estimation using a discretized adaptive multi-regional approach. This is, indeed, equivalent to approximating the unknown function by a "patchwork" (lattice) of intra-regional and inter-regional explicit subfunctions all of which operate on a grid with a user-defined resolution. In all of the works previously reported, the subregions are selected *a priori* based on subjective judgements and not subsequently modified [3, 16]. However, in the method proposed in [5, 46], the region of interest is subdivided into a set of sub-regions adaptively using a VQ method, and in our current work this has been done by *only* restricting ourselves to "integer" points on the grid. Both of these impose an implicit discriminant mapping on the domain. Subsequently, the arbitrary distance function is sub-classified as a set of intra-set and inter-set distance functions each of them being characterized using their own respective parameters. In each case the training sites and their corresponding available distances are then used to train the intra and inter-set parameters whence the estimation follows. All of these ideas are novel to the area of distance estimation. But we believe that the fundamental highlight of our contribution from a conceptual perspective is the application of merging of multiple learning paradigms in the current application domain.

Most of the research that is currently available in distance estimation involves the estimation of geographical road travel distances. Consequently, to place our current work in the right perspective, in Section 2 we shall review the currently available results in distance estimation as applicable to this domain. In Section 3 we shall give an overview of LA and the advantages of discretization. In section 4 we shall explain the concepts of VQ and the SOM and proceed to show how they can be applied to the estimation of arbitrary distance functions. Section 5 discusses the experimental results and highlights the salient features of our methods in the context of both the optimization and neural network strategies. Section 6 concludes the paper.

2 Road Travel Distance Estimation

2.1 Distance Estimation Problem

The *actual distance* between any two points on the earth surface is the length of the shortest road connecting them. Since it is often not feasible to measure the *actual distances* for all pairs of points, it is a common practice to use distance estimators. Then the question is to choose a good estimator so that accurate distance approximations are obtained.

A good estimation of actual distances is critical in many applications. Almost all of the location problems, distribution problems such as the transportation problem, its generalization the transshipment problem, the traveling salesman problem, and the vehicle routing problem assume the knowledge of actual

[1] In the U.S. the actual distances are readily available. The applicability of our technique for arbitrary distance function estimation is *demonstrated* using Road distances in Türkiye. But in a more general setting these concepts can be used in evaluating distances in arbitrary spaces - for example in computing distances between macromolecules or even in cortical maps. We are currently investigating this. We are grateful to an anonymous referee of our previous work who pointed this out to us.

distances in their formulations. For example, in their simulation study to determine the number of fire-stations in İstanbul, Erkut and Polat multiply the Euclidean distance by an inflation factor, which they call the *road coefficient*, in order to estimate the actual distance between the fire-station and fire area [15].

We can define the problem of distance estimation formally as follows: Let us say that P_a and P_b are two points on the Cartesian plane with coordinates $P_a = (x_{a1}, x_{a2})^T$ and $P_b = (x_{b1}, x_{b2})^T$. The aim is to build an estimator $\Phi(P_a, P_b|\boldsymbol{\theta})$ of the actual distance between P_a and P_b. Let $\pi_i = \langle P_{i1}, P_{i2} \rangle$ be the ith **pair** of points, and let r_i be the actual distance between P_{i1}, P_{i2}. The set of all pairs and the corresponding distances is given by S as :

$$S = \{(P_{i1}, P_{i2}, r_i) : 1 \leq i \leq n\}, \; where, \quad n = \binom{N}{2} \tag{1}$$

Here N and n are respectively the number of points and pairs formed by using them. $\boldsymbol{\theta}$ is a vector of parameters estimated using S with respect to the following goodness-of-fit criterion :

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left[E[\delta(\Phi(P_{i1}, P_{i2}|\boldsymbol{\theta}, S), r)]\right] = \arg\min_{\boldsymbol{\theta}} \left[\frac{1}{n} \sum_{i=1}^{n} \delta\left(\Phi(P_{i1}, P_{i2})|\boldsymbol{\theta}, r_i\right)\right] \tag{2}$$

$\delta(\cdot)$ is the difference measure. One possibility, originally proposed by Love and Morris [30], is the absolute value of the deviation:

$$\delta\left(\Phi(P_{i1}, P_{i2}|\boldsymbol{\theta}), r_i\right) = |\Phi(P_{i1}, P_{i2}|\boldsymbol{\theta}) - r_i| \tag{3}$$

According to this criterion, a distance function must estimate greater actual distances relatively more accurately than shorter distances. This is a drawback if we are more interested in proportional deviations than absolute deviations. Another error measure, also proposed by Love and Morris [30], is normalized by dividing pairwise estimation errors by the square root of the actual distance between them :

$$\delta\left(\Phi(P_{i1}, P_{i2}|\boldsymbol{\theta}), r_i\right) = \left[\frac{\Phi(P_{i1}, P_{i2}|\boldsymbol{\theta}) - r_i}{\sqrt{r_i}}\right]^2 \tag{4}$$

Although both criteria provide ample insight in their own right, the latter one is superior not only because it gives importance to proportional errors but also because of the following three reasons. First, most of the experimental results in the literature use the second criterion, e.g., [4, 6, 12, 30, 34, 59] and hence serve as an excellent benchmark. Furthermore, it has important statistical properties which leads to statistical tests for comparing the accuracy of distance functions under certain normality and independence assumptions, and thus the results obtained can be statistically justified. Finally, it is a continuous and differentiable function of the parameter vector which enables the use of gradient descent minimization strategies important in various domains including neural network learning.

The standard approach for distance estimation uses estimators that are parameterized functions of certain "easy-to-obtain" pieces of information, namely the coordinates of the points. This approach has been widely used ever since the first work by Love and Morris [30] because it provides simple analytical closed form expressions of the coordinates once the values of the parameters have been determined. As in any parametric method, the concept works well with small samples, but the accuracy may not be high if the assumed form of the function is not appropriate.

In the recent work by Alpaydın *et. al.* [2] the problem of estimating distances has been viewed in the context of function approximation or nonlinear regression, and perceptron based estimators have been applied for this task of estimating $\Phi(P_{i1}, P_{i2}|\boldsymbol{\theta})$. These methods, being nonparametric, have the advantage that they do not assume any *a priori* model and are trained directly from a training sample. They, of course, necessitate larger training samples and more computer time as the simplicity of a parametric model with just a few parameters does not exist anymore. Although, perceptron based non-parametric estimators perform better compared to parametric distance functions, (i.e., they yield smaller errors), the results can be improved further if the cities are clustered adaptively using a VQ [5, 46] or DVQ method prior to any estimation attempt. Indeed, as it can be philosophically justified, VQ and DVQ are hybrids between the parametric and non-parametric families of algorithms.

Table 1: Distance functions used and their associated parameters.

DISTANCE FUNCTION	PARAMETERS (θ)
$\Phi_1(P_1,P_2) = k(\lvert x_{11}-x_{21}\rvert + \lvert x_{12}-x_{22}\rvert)$	k
$\Phi_2(P_1,P_2) = k(\lvert x_{11}-x_{21}\rvert^2 + \lvert x_{12}-x_{22}\rvert^2)^{1/2}$	k
$\Phi_3(P_1,P_2) = k(\lvert x_{11}-x_{21}\rvert^p + \lvert x_{12}-x_{22}\rvert^p)^{1/p}$	k,p
$\Phi_4(P_1,P_2) = k(\lvert x_{11}-x_{21}\rvert^p + \lvert x_{12}-x_{22}\rvert^p)^{1/s}$	k,p,s

2.2 Distance Functions

A generally used method for estimating actual distances between any pair of points is to make approximations by means of a distance function, which is a parameterized function of the planar coordinates of the two points. These functions can be classified in three major groups with respect to the type of coordinates they use. The members of the first group use spherical coordinates for the purpose of introducing the spherical effect of the earth surface into the distance estimation [30, 31]. Although this idea provides certain additional accuracy, the contribution has been experimentally reported to be minor by Love and Morris [30]. The second group consists of functions which use polar coordinates [39, 50]. The motivation is based on the observation that the roads in historically older cities are not usually planned according to a rectangular grid structure and consequently, distances are often better approximated by a ring-radial measure. This approach seems to be very accurate especially for a spider's web-like road network structure. The third group contains some simple functions of the Cartesian coordinates. These are mostly norms or norm-based functions, and the ones we have adopted are listed in Table 1. Indeed, in the literature these are the most important ones, because of their wide usage in location and distribution problems [17, 33].

The parameters, which should be nonnegative, k, p, and s, constitute θ, and are estimated over the sample to provide good approximations and as such, encode geographical characteristics of the region where they are used. There is a large literature on the determination of these parameters and the comparison of the parametric distance functions. Astonishingly enough, some of the conclusions drawn in these papers are conflicting [6, 7, 8, 10, 30, 31, 32].

For all practical purposes, the function chosen to estimate actual road distances should be as accurate as possible. In their early study, Love and Morris [30, 31] compute the parameters k, p, and s of $\Phi_1(P_1,P_2), \Phi_2(P_1,P_2), \Phi_3(P_1,P_2)$, and $\Phi_4(P_1,P_2)$ for the United States and compare them with respect to the accuracy they provide. The important conclusion of this study is the superiority of $\Phi_4(P_1,P_2)$ over the other three. The second best approximating function seems to be $\Phi_3(P_1,P_2)$.

At the end of their study on the road network of the former Federal Republic of Germany (FRG), Berens and Körling [7] and Berens [6] conclude that the accuracy provided by the weighted Euclidean norm $\Phi_2(P_1,P_2)$ is sufficient and the use of $\Phi_3(P_1,P_2)$ is not worth the extra computational effort necessary for calculations. However, in a further study over the largest 25 cities of FRG, Love and Morris [32] report conflicting results which demonstrate that the accuracy of the weighted L_p norm, $\Phi_3(P_1,P_2)$, is remarkably higher than the accuracy provided by $\Phi_2(P_1,P_2)$. Although it supports the early findings of Berens and Körling [7] for FRG, the study by Berens [6] includes mixed results when it is enlarged to cover 11 other countries; the relative improvement introduced by $\Phi_3(P_1,P_2)$ over $\Phi_2(P_1,P_2)$ ranges within 0.00% and 11.27%. Finally, Berens and Körling [8], in their last comment, state that, if the accuracy is of primary interest, the empirical distance functions should be tailored for the regions they are to be used for. Currently, there is no single general distance function which provides the same accuracy all over the world.

There are also distance measures which do not fit completely in any of the above mentioned three groups. They can be included in the last category, but they are not always simple functions of the coordinates and require additional information such as a rotating angle for the coordinate axes [12, 31] or vectors for possible directions on a typical road [59, 60]. All of them are based on the idea that a travel has two major components; rectilinear and Euclidean, and the actual distance between any pair of points

can be modeled as their non-negative linear combination. Ward and Wendell [59] initiate this hybrid idea by suggesting the weighted one-infinity norm and observe that the accuracy of this function is relatively close to the accuracy of the weighted L_p norm, $\Phi_3(P_1, P_2)$, based on the data set of Love and Morris [30]. In their later work, Ward and Wendell generalize the one-infinity norm to obtain the family of block norms in which the accuracy of the approximation depends on possible travel directions [60]. They report that the approximations obtained by the weighted L_p norm are more accurate than those obtained by a two-parameter block norm, which is actually the weighted one-infinity norm, and the accuracy of the weighted L_p norm is slightly worse than the one of eight-parameter block norms. Similar conclusions have been obtained also by Love and Walker [34] in their detailed empirical study on block and round norms. Block norms play an important role in location models because they lead to linear programming problems for certain objective functions, such as the minimax distance function; but the size of the linear program can easily become very large.

Another hybrid distance function is due to Brimberg and Love [11]. It is called the weighted one-two norm since the rectilinear and Euclidean elements of the travel are presented respectively by the weighted L_1 and L_2 norms. The authors suggest its use to approximate $\Phi_3(P_1, P_2)$ in estimating distances. The weighted one-two norm provides also good approximations for the probabilistic L_p norm [13]. Besides, its parameters can be calculated easily by simple linear regression [10], and it can perform very well when local information is also introduced through the rotation of coordinate axes.

Due to the statistical nature of distance functions, the unknown distance between the points may be overestimated or underestimated. Then, confidence intervals for unknown distances become important since they can be used to measure the accuracy of the estimated distance. In the recent work of Love et al. [35], this issue has been addressed. They have developed a procedure for calculating confidence intervals for unknown distances. Their procedure utilizes information provided by the sample Pearson coefficients.

Having briefly surveyed the field we are now in a position to explain how VQ and its discretized version, DVQ, can be incorporated in an adaptive multi-regional strategy, and applied to the estimation of arbitrary distance functions.

3 Learning Automata, and Discretized Vector Quantization

3.1 Learning Automata

Variable Structure Stochastic Automata (VSSA) were developed by Varshavskii and Vorontsova. For these automata, the learning process is generalized so that the state transition probabilities and the action selecting probabilities evolve with time [41]. The automaton is simplified in the sense that each state now corresponds uniquely to a particular action. Hence while in state ϕ_i the automaton always picks one action α_i from a finite set A or r actions, and consequently, the set of states is redundant. Thus, what remains is the set of actions (or output from the automaton), the set of inputs (one of which serves as the input to the automaton at any time instant) and a learning algorithm T. The learning algorithm operates on a probability vector $P(t)$ whose i^{th} component $p_i(t)$ is the probability that the automaton will select action α_i at time t, with the components summing to unity. Indeed, if B is the set of inputs and A the set of actions, the learning algorithm is completely defined by a function T such that $T(P(t), A(t), b(t)) = P(t+1)$. Many varieties of absorbing and ergodic VSSA have been documented [28, 41]. In both cases they can be made to converge to the optimal action with a probability as close to unity as desired.

3.2 Discretized Learning Automata

The beauty of a discrete learning algorithm is that it does not ignore the limitations of practical implementations ; on the contrary this limitation is used to its advantage. VSSA evolved from fixed structure stochastic automata as an attempt to simplify the analysis of the automata's properties [28, 41]. However, VSSA have a limitation. Implicit in the definition of VSSA is the fact that the probability of choosing an action can be any real number in the interval [0, 1]. Rendering this probability space discrete is a general approach for improving VSSA [43, 44, 45, 47]; this is implemented by restricting the probability of choosing an action to only finitely many values from the interval [0, 1]. Consequently, probability changes

are made in jumps and not continuously. In a sense, the discrete VSSA represent a hybrid of a fixed structure automaton and VSSA.

Discrete automata consist of finite sets like Finite Structure Stochastic Automata, but they are VSSA because they are characterized by a probability vector which evolves with time. Discrete algorithms are linear if the probability values are equally spaced in the interval [0, 1]; otherwise, they are called non-linear [43]. Existing literature [43, 44, 47] uses the term "discretized" in front of the name of a learning automaton to indicate the discrete version of a continuous VSSA. The history of discretized automata (which ignore and use estimates) and the various reported families and their asymptotic properties are catalogued in [44, 47].

Probably the biggest limitation of learning automata is their slow rate of convergence [28, 41]. By limiting the number of assumptions that learning automata have about the environment, they are a general approach for machine learning. However, this also means that there are fewer properties that can be used to speed up the rate of convergence. Originally the intent of introducing discrete learning automata was to increase the rate of convergence and to eliminate the assumption that the random number generator could generate real numbers with arbitrary precision [43, 44, 47]. Once the optimal action has been determined, and the probability of selecting that action is close to unity, discrete automata increase this probability directly, rather than approach the value unity asymptotically. Indeed, by making the probability space discrete, a minimum step size is obtained. If the automaton is close to an end state, the minimum step size forces it to this state with just a few more favourable responses.

The central issue from a theoretical point of view is that the properties of a Markov process can change if the probability of choosing an action is restricted to a finite subset of [0,1]. For example, a continuous space will have recurrent states, but a finite space will only have positive recurrent states [41]. As well, discrete Markov processes have properties that are not true for general Markov processes. Round off error will cause the automaton that approaches its end point asymptotically to artificially reach its end point [43, 44, 47]. Also, the proofs of convergence in continuous spaces may not be applicable to a finite state machine. This point is demonstrated by the fact that, so far, the existing proofs of convergence for discrete algorithms are significantly different from the proofs of their continuous counterparts (compare [43, 44, 47] to the methods used in [28, 41]).

As alluded to earlier, another benefit of discretizing the probability of choosing an action is that it reduces the requirements on the system's random number generators. This is important since VSSA use a random number generator in its implementation [28, 41]. In theory, it is assumed that any real value in [0, 1] can be obtained from the machine; in practice, only a finite number of these values are available.

Finally, and far from being unimportant are the considerations of implementation and representation. Discrete versions lead, quite naturally, to the use of integers for keeping track of how many multiples of the resolution parameter the action probabilities are. While the above consideration frequently increases the rate of convergence measured in terms of the number of iterations, a discrete algorithm also has the benefit of reducing the time measured in terms of the clock cycles that a microprocessor would take to do each iteration of the task. It also reduces the amount of memory needed. Typically addition is quicker than multiplication on a digital computer, and the amount of memory used for a floating point number is usually more than that required for an integer. In the schemes that have been discretized so far, whereas the continuous versions update their probability vectors via multiplication, the discretized counterparts achieve this with addition and subtraction. Thus, in terms of both time and space, discrete algorithms seem to be superior.

3.3 Analog to Digital Conversion

Neural networks modify their weights and the effect of their inputs by "output" functions which are often sigmoidal. As opposed to using traditional output functions, the entire concept of discretization can be *perceived*[2] as a way by which the output of the machine is constrained to take a value which is on a finite grid whose resolution is J_i on the i^{th} axis. Effectively, this means that we resort to "Analog-to-Digital" (A_To_D) converters each of which has as its input a real value, and which yields as its output an integer value in $[0, ..., J_i]$ which is the discretized "grid" coordinate of the point concerned in the respective

[2] This is not how a discretized automaton is defined. But if it is viewed conceptually from this perspective it permits us to view the present adaptation of VQ in a more pragmatic manner.

direction. Thus, central to the process of discretization is the function "*A_To_D*" which achieves this. The function has parameters which are the sizes of the grids on the axes, say, $GridSize$. The function itself (the details of which are obvious and consequently omitted here) has as its input a point on the Cartesian plane P with coordinates $P = (x_1, x_2)^T$. It yields as its output the discretized version of the point, $P^d = (x_1^d, x_2^d)^T$ where the components are integers and given by:

$$x_1^d = Round(x_1/GridSize)$$

and,

$$x_2^d = Round(x_2/GridSize).$$

To explain how the Discretized VQ is achieved we shall describe how the Inter and Intra-regional polarizing concepts of VQ are effected in the continuous domains and how they are extended to the discretized domain using the analogue-to-digital conversion described above.

3.4 Vector Quantization

The foundational ideas motivating VQ and the SOM are classical concepts that have been applied in the estimation of probability density functions. Traditionally, (in the realms of both statistical analysis and statistical pattern recognition) distributions have been represented either parametrically or non-parametrically. In the former, the user generally assumes the form of the distribution function and the parameters of the function are learnt using the available data points. In pattern recognition (classification), these estimated distributions are subsequently utilized to generate the discriminant hyperspheres (or hyperellipsoids) whence the classification is achieved.

As opposed to the latter, in non-parametric methods, the practitioner assumes that the data must be processed in its entirety (and not just by using a functional form to represent the data). The corresponding pattern recognition (classification) algorithms which result are generally of the nearest neighbor (or k-nearest neighbor) philosophy and are thus computationally expensive. The comparison of these two perspectives is found in standard pattern recognition textbooks [14, 18], and bounds on the classification error rate of non-parametric strategies (as compared to the optimal Bayesian) parametric strategies have also been derived.

The concept of VQ can be perceived as a compromise between the above two schools of thought. Rather than represent the entire data in a compressed form using only the estimates (and in the estimate domain), VQ opts to represent the data in the actual feature space. However, as opposed to the non-parametric methods which use all the data in the training and testing phases of classification, VQ compresses the information by representing it using a "small" set of vectors, called the code-book vectors. These code-book vectors are migrated in the **feature** domain so that they collectively represent the distribution under consideration. We shall refer to this phase as the *Intra-Regional Polarizing* phase. In a multi-class problem the code-book vectors for each region are subsequently migrated so as to ensure that they adequately represent their own regions and furthermore distinguish between the other regions. This phase, which we refer to as the *Inter-Regional Polarizing* phase, also implicitly learns the discriminant function to be used in a subsequent classification module. Note that these discriminant functions are of a nearest neighbor philosophy, except that the nearest neighbors are drawn from the set of code-book vectors (as opposed to the entire set of training samples). They thus drastically reduce the computational burden incurred in the testing of traditional non-parametric methods. It is not appropriate that we explain the details of VQ and the SOM here; they can be found in an excellent survey by Kohonen [24] and in [25]. However, in the interest of completeness and continuity, we shall **in all brevity**, explain the various phases of the VQ modules.

3.5 Intra-Regional Polarizing

We assume that we are to estimate the distance $\Phi(P_j, P_m)$ between any two points P_j, P_n in the set of points G. We also assume that we are given (the training set) L, a subset of G and the inter-node distances for the nodes in L (i.e., $\{\Phi(P_j, P_m) | P_j, P_m \in L\}$).

The basic hypothesis in distance estimation using a multi-regional approach is that G can be partitioned into a set of smaller regions whence intra-regional and inter-regional approximates of Φ can be obtained. Thus, in the training phase [3], we partition L into W subsets,

$$C_k = \{P_{k,i} : 1 \leq i \leq N_k\}(1 \leq k \leq W). \tag{5}$$

Our primary aim is to represent each C_k by M representative points ($M \ll N_k$) [4] $\{Q_{k,j} : 1 \leq i \leq M\}$. The set of code-book vectors $\{Q_{k,j} : 1 \leq j \leq M\}$ are first initially randomly assigned positions within or close to their respective regions. In the intra-regional polarizing the algorithm is repeatedly presented with a node $P_{k,i}$ from C_k^d. The closest code-book vector, $Q_{k,j}$, to $P_{k,i}$ is determined and this vector is moved in the direction of this data point. Indeed, this is achieved by rendering the new $Q_{k,j}$ to be a convex combination of the current $Q_{k,j}$ and the data point $P_{k,i}$. More explicitly, the updating algorithm is as follows :

$$Q_{k,j}(t+1) = \begin{cases} (1-\alpha)Q_{k,j}(t) + \alpha P_{k,i} & if \;\; Q_{k,j} \;\; is \;\; the \;\; closest \;\; point \\ & to \;\; the \;\; data \;\; point \;\; P_{k,i} \\ Q_{k,j}(t) & otherwise \end{cases} \tag{6}$$

where 't' is the discretized (synchronized) time index.

We now consider how these concetps can be extended to a discretized philosophy. To discretize things, in the training phase, we project all the training points onto the grid by repeatedly invoking A_To_D on them. Thus, we have R subsets of *discretized* training partitions C_k^d, where, L is partitioned into R subsets,

$$C_k^d = \{A\_To\_D(P_{k,i}) : 1 \leq i \leq N_k\}(1 \leq k \leq R). \tag{7}$$

Again we represent each C_k by M representative *discretized* points ($M \ll N_k$) $\{Q_{k,j}^d : 1 \leq i \leq M\}$. The set of code-book vectors $\{Q_{k,j}^d : 1 \leq j \leq M\}$ are first initially randomly assigned positions within or close to their respective regions, but constrained to be on the grid themselves by invoking A_To_D to their random real representations. Thus, in the intra-regional polarizing the algorithm is repeatedly presented with a node $P_{k,i}^d$ from C_k^d. The closest code-book vector, $Q_{k,j}^d$, to $P_{k,i}^d$ is determined and this vector is moved in the direction of this data point.

Since we are working in a discretized space, we have to consider what we mean by the "closest" code-book vector. Indeed, the more fundamental question is one of determining how distances will be measured in this space [24]. Since the primary intention in working in a discretized space is to work with integers (and to minimize real computations) the distance used in this case is what we call the "Discretized Eucledian" (E^d) which approximates the distance between pixels points if traversed along the pixel directions in a two-dimensional pixel array. Let us now consider how we can move from pixel P_a^d to P_b^d.

If $P_a^d = (x_{a,1}^d, x_{a,2}^d)^T$ and $P_b^d = (x_{b,1}^d, x_{b,2}^d)^T$ are two discretized points on the grid we see the number of pixels to be traversed in the two directions is the difference of their respective coordinates given by $Diff1$ and $Diff2$ as :

$$Diff1 = x_{a,1}^d - x_{b,1}^d \;, \; and,$$

$$Diff1 = x_{a,2}^d - x_{b,2}^d.$$

It is clear that the minimum number of diagonal pixel traversal which has to be done is given by $DiagMoves$, where,

$$DiagMoves = Min\{Diff1, Diff2\}. \tag{8}$$

Once these diagonal moves have been achieved, the linear moves to be done to go from P_a^d to P_b^d is given by $LinearMoves$, where,

$$LinearMoves = Max\{Diff1 - DiagMoves, Diff2 - DiagMoves\}. \tag{9}$$

[3] In what follows, as opposed to the notation of Section 2.1, $P_{k,i}$ will represent the ith point in the kth region.

[4] Although strictly speaking, we can represent a set C_k by M_k points (where M_k increases with N_k), in the interest of simplicity, in this paper we have assumed that the number of representative code-book points for all the classes is the same.

Thus, the Discretized Eucledian Distance (E^d) between P_a^d and P_b^d is :

$$E^d(P_a^d, P_b^d) = \sqrt{2} \cdot DiagMoves + LinearMoves. \tag{10}$$

It is clear that the computation of the Discretized Eucledian Distance (E^d) requires *only five* integer additions (subtractions), two comparisons and a single multiplication.

Since the distance function is a linear combination of the distances traversed in the individual dimensions, minimizing in each direction would minimize the overall distance. Thus, arguing as in [24], we can update the code-book vectors by setting $Q_{k,j}^d$ to be a convex combination of the current $Q_{k,j}^d$ and the discretized data point $P_{k,i}^d$ and projecting back onto the discretized space after computation. Consequently, the new updating algorithm is as follows :

$$Q_{k,j}^d(t+1) = \begin{cases} A\_To\_D\left((1-\alpha)Q_{k,j}^d(t) + \alpha P_{k,i}^d\right) & \text{if } Q_{k,j}^d \text{ is the closest point} \\ & \text{to the data point } P_{k,i}^d \\ Q_{k,j}^d(t) & \text{otherwise.} \end{cases} \tag{11}$$

Note that this can be seen to be the discretized version of the traditional SOM strategy [21, 24, 25, 29, 36] except that we have (as in [5, 46]) consistently restricted the radius of the "bubble of interest" used by Kohonen to be unity. The reasons for this are two-fold :

1. Since we are attempting to represent the nodes in C_k by a set of representative code-book vectors, the *topological ordering* of these code-book vectors is absolutely irrelevant. This, in turn, makes the algorithm computationally extremely inexpensive, because, at each time we need only locate the nearest code-book vector - using simple *integer computations*, and do not have to find **all** the code-book vectors within the bubble of activity.

2. In a typical application, the number of code-book vectors must be kept **extremely small**. This is because, we want to partition G into R sub-regions and thus, effectively, we are attempting to approximate function Φ using $(M \times R) \times (M \times R - 1)/2$ "patched" functions. If each of these functions has 3 parameters, the number of parameters to be estimated becomes prohibitively large. Thus, if R is 4 and M is 3, the number of parameters is 234. Indeed, if we represented each region by $M = 4$ code-book points, the number of parameters involved would be 408. Indeed, considering the extremely small values of M encountered in this application domain, (we have used $M = 3$ per region) rendering the radius of the "bubble" of activity to be unity is far from unreasonable. Furthermore, as mentioned above, it only hastens the rate of convergence of the scheme.

In (11) above, we have decremented α linearly from unity for the initial learning phase and then switched to small values of α which decrease linearly from 0.2 for the fine-tuning phase. This is as recommended in the literature [24, 25] and has been justified in the continuous domain [5, 46]. Each region is subjected to the intra-regional polarizing before the next phase, the inter-regional polarizing is invoked.

3.6 Inter-Regional Polarizing

After the individual regions have been represented by a subset of M code-book vectors using the above migration strategy, the code-book vectors are tested using L to see whether they adequately classify the points within their respective sub-regions. To achieve this, we resort to an algorithm analogous, in principle, to the LVQ3 algorithm [24]. Every data point in the test set, L (not just in the individual clusters, C_k), is tested against the set of Q_k's to see whether their nearest code-book vector falls within their partition. Thus, unlike the previous phase, where the code-book vectors found their respective places by learning only from the location of the data points **within their** own respective classes, in this phase, these representative vectors are migrated so that they polarize **away from** the data points of the **other competing clusters**. The principle by which this is done in the continuous world is as follows.

Let us suppose that we examine a point $P \in C_k$. Also let us suppose that the two closest code-book vectors to P (among **all** the $\{Q_{k,i}\}$) are Q_a and Q_b. If both Q_a and Q_b do not belong to the cluster C_k, clearly, the information content in P (with respect to Q_a and Q_b) is misleading, and so it is futile

to migrate Q_a and Q_b using this information. However, if both of them are intended to represent C_k, clearly, the information in P can be used to achieve an even finer tuning to their locations. Thus, in this scenario, both Q_a and Q_b are moved marginally from their current locations along the hyperline **towards** P. The final scenario is the case when one of them, Q_a (Q_b), correctly belongs to C_k, and the other, Q_b (Q_a), belongs to a different partition. In this case, the information in P can be used to achieve an even finer tuning to their locations by migrating Q_a (Q_b) marginally from its current location along the hyperline **towards** P, and migrating the other code-book vector Q_b (Q_a) marginally from its current locations along the hyperline **away from** P.

Since we do not want the "straggler" points (the points which are misclassified, but which probably would not have been correctly classified even by an optimal classifier) to completely dictate (and thus, disturb) the polarizing, this migration is invoked only if the node P lies within a pre-specified window of interest, W. This restriction has also been recommended in the literature [24, 25], and typically, this window, W, is a hypersphere centered at the bisector between the codebook vectors Q_a and Q_b. Also, as recommended in the literature, the polarizing of **both** Q_a and Q_b (when both of them correctly classify P) is made to be of much smaller magnitude than in the scenario when either of them misclassifies it. These steps are formally given in [5, 46] for the continuous world.

In the discretized world the modifications are done by performing all the migrations mentioned above on the grid with the additional constraint that "distances" and "nearest neighbours" are evaluated using the discreized Eucledian distance $E^d(\cdot)$. Thus the modified inter-regional polarizing equations are as follows.

If Q_a^d and Q_b^d are the two closest code-book representative vectors to a given point $P^d \in C_k^d$

$$\begin{array}{rcll}
Q_a^d(t+1) & = & (1-\epsilon\alpha)Q_a^d(t) + \epsilon\alpha P^d & if \;\; Q_a^d, Q_b^d \in C_k \;;\; P^d \in W \\
Q_b^d(t+1) & = & (1-\epsilon\alpha)Q_b^d(t) + \epsilon\alpha P^d & if \;\; Q_a^d, Q_b^d \in C_k \;;\; P^d \in W \\
Q_a^d(t+1) & = & (1-\alpha)Q_a^d(t) + \alpha P^d & if \;\; Q_a^d \in C_k \;;\; Q_b^d \in C_j \neq C_k \;;\; P^d \in W \\
Q_b^d(t+1) & = & (1+\alpha)Q_b^d(t) - \alpha P^d & if \;\; Q_a^d \in C_k \;;\; Q_b^d \in C_j \neq C_k \;;\; P^d \in W \\
Q_a^d(t+1) & = & (1+\alpha)Q_a^d(t) - \alpha P^d & if \;\; Q_a^d \in C_j \neq C_k \;;\; Q_b^d \in C_k \;;\; P^d \in W \\
Q_b^d(t+1) & = & (1-\alpha)Q_b^d(t) + \alpha P^d & if \;\; Q_a^d \in C_j \neq C_k \;;\; Q_b^d \in C_k \;;\; P^d \in W \\
Q_a^d(t+1) & = & Q_a^d(t) \;;\; Q_b^d(t+1) = Q_b^d(t) & otherwise
\end{array} \tag{12}$$

In (12) above, we have maintained α at a constant value of 0.1 (as opposed to varying it as recommended in [24, 25]), and kept ϵ to be 0.25. Also, the window W defined above was set to be a circle of diameter 1/100 of the distance between $Q_a^d(t)$ and $Q_b^d(t)$.

As in the continuous case, the reader should observe that after the intra-regional polarizing and the inter-regional polarizing, the representative code-book vectors impose a set of piece-wise linear boundaries which assign the original nodes, L, into potentially slightly different regions than that which was initially assigned. Thus, although the initial demarcation boundaries may have been incorrectly assigned, the sequence of polarizing operations tends to re-allocate them. The effect of this boundary re-allocation will be discussed in greater detail in the section describing our experimental results.

After the training points have been allocated and the set of code-book vectors for each cluster has been learnt, the patchwork of functions approximating Φ is now learnt. This can be done using either an independent optimizing strategy or a VQ scheme. We shall now demonstrate how these are achieved.

3.7 Parameter Learning Using VQ

After the individual regions have been represented by the various *discretized* codebook vectors (using the above migration strategies), we are now in a position to estimate Φ. The basic assumption in this phase, is that we can approximate Φ by a patchwork (or lattice) of intra-regional and inter-regional functions. In this phase, we shall attempt to learn these respective approximating functions using the code-book vectors and the given (true) distances $\Phi(P_i, P_j)$, P_i, $P_j \in L$.

Let P_i and P_j be any two nodes in G. Obviously, if only the points in G are of interest to us, Φ can be approximated (indeed, exactly represented) in terms of all the inter-node Euclidean norms as follows :

$$\Phi(P_i, P_j) = k_{i,j}||P_i - P_j|| \tag{13}$$

Notice that since Φ is symmetric, only roughly half of these coefficients will have to be estimated. Clearly, such a representation defeats the fundamental purpose of a distance estimation strategy, for it would necessitate the learning of all the $\{k_{i,j} : 1 \leq i, j \leq N; i > j\}$ coefficients. Our intention is to approximate (13) by hypothesizing that the constants $\{k_{i,j}\}$ are dependent on the code-book representative vectors. Thus, rather than specify $\Phi(P_i, P_j)$ using (13) above, we assume that $\Phi(P_i, P_j)$ can be reasonably approximated by locating the closest code-book vectors for P_i and P_j and evaluating a simple function between these respective points. Thus, we approximate $\Phi(P_i, P_j)$ by using (14) below :

$$\Phi(P_i, P_j) \cong k_{a,b} \parallel P_i - P_j \parallel \tag{14}$$

where, the closest code-book to P_i^d and P_j^d are Q_a^d and Q_b^d respectively. The problem that is now before us is one of determining the set of parameterizing constants $\{k_{a,b} : 1 \leq a, b \leq R \times M; a \leq b\}$.

There are at least three distinct schemes for evaluating the above set of parameterizing coefficients, $\{k_{a,b}\}$ using the training set, L and their corresponding true distances.

The first is by a simple averaging strategy. For every pair of nodes in the training set, a cumulative sum of the ratio of their true distance to their Euclidean distance is maintained. This ratio is called the *directional bias* by Brimberg and Love [11] and Brimberg and Wesolowsky [13]. This sum is associated with the pair of closest code-book vectors. The cumulative sum divided by the number of pairs using these code-book vectors yields the average value of $k_{a,b}$ for these code-book vectors Q_a^d and Q_b^d.

An alternate strategy to obtain the set of parameterizing coefficients is to perform a VQ learning algorithm in the space involving the coefficients themselves. We explain this strategy as follows. Let us suppose that we have a current value of $k_{a,b}$. When a new pair of points in L is examined, if the codebook vectors are Q_a^d and Q_b^d respectively, the updated value of $k_{a,b}$ is obtained by moving the current value towards the value estimated using just this set of points. This updating is done along the hyperline joining the two. This is formally described below.

Algorithm : GetCoeffByVQ
Input : The set of codebook vectors, the training set L and the distances $\Phi(P_i, P_j)$ for all P_i, $P_j \in L$.
Output : The set of parametizing coefficients, $\{k_{a,b} : 1 \leq a, b \leq R \times M; a \leq b\}$.
Begin
  For $a = b$ to $R \times M$ **Do**
    $\lambda_{a,b} = 1$
    $k_{a,b} = 0$
  EndFor
  Repeat until satisfied
    Get any distinct pair of points P_i, P_j in L
    If Closest code-book vectors to P_i^d and P_j^d are Q_a^d and Q_b^d respectively **Then**
      $k_{a,b} = (1 - \lambda_{a,b})k_{a,b} + \lambda_{a,b}(\Phi(P_i, P_j)/ \parallel Q_a - Q_b \parallel)$
      Decrease $\lambda_{a,b}$
    EndIf
  EndRepeat
End GetCoeffByVQ

It is easy to see that if $\lambda_{a,b}$ decreases inversely with the number of samples encountered (which have Q_a^d and Q_b^d as the code-book vectors) the above VQ strategy converges **exactly** to the average value of $k_{a,b}$ computed earlier. Any other updating method for $\lambda_{a,b}$ would converge to an alternate (hopefully, closer-to-optimal) value of the $k_{a,b}$. In all our experiments, we have used an inverse decreasing function for $k_{a,b}$. Indeed, in this setting, our results tend to show that, (as opposed to the speech recognition example discussed in [24]) we now have a scenario when an all-neural approach ([24] pp.82) is recommendable.

The third approach involves explicit optimization. Here, each inter-node distance is specified by a function which is completely defined by the closest code-book vectors, and whose functional form is one of those types tabulated in Table 1. The question of estimation is now reduced to one of optimization as has been done in the literature [2, 30, 31]. Here, Kohonen's recommendation of using a traditional scheme subsequent to a neural strategy has proven to be superior.

4 IMPLEMENTATION RESULTS

4.1 Database

We have collected two distinct samples by pairing respectively 80 and 23 cities and towns of Türkiye for training and test sets. The first is used for estimating the parameters and the second one to assess their performances. The training and test data sets contain planar coordinates and intercity distances for 80 and 23 cities respectively which make 3160 and 253 data pairs. The third dimension is ignored since previous empirical studies have shown that the effect of elevation in the accuracy of the estimators in Türkiye is almost null [4]. The values we report in the following sub-sections are average error per pair in kilometers on both the training and the test sets. Recall that the error per pair is measured by the normalized error measure given in Equation (4).

4.2 Distance Functions

By looking at properties of the application, it is realistic to eliminate some of the distance functions *a priori* by judging them with the structural properties of the actual road network. First of all, the road structure in Türkiye has been developed arbitrarily rather than rectilinearly or ring-radially. This arbitrariness makes the identification of a fixed pattern for possible travel directions within the country impossible; this is crucial for the use of hybrid norms. Besides, the area is too small to require the consideration of the earth's roundness in estimating actual distances. Being convinced by these observations, it is rational to concentrate on functions $\Phi_2(P_a, P_b), \Phi_3(P_a, P_b), \Phi_4(P_a, P_b)$ and compute the best possible value of the parameters k, p, and s. In spite of this fact, we also computed the value of k for $\Phi_1(P_a, P_b)$, since it has been heavily considered in the related literature [17, 33].

The calculation of the parameters with respect to any of the goodness-of- fit criterion introduced in Section 1 requires minimizing an error function similar to that of Equation (2). Since we use the normalized error function given in Equation (4), the minimization problems are continuous in the parameters. Karush-Kuhn-Tucker first order conditions for the first two problems have analytical solutions and the values of k which minimize error function (4) can be easily obtained by using the following equalities:
For $\Phi_1(P_{i1}, P_{i2})$:

$$k = \frac{\sum_{i=1}^{n} \left(|x_{i,11} - x_{i,21}| + |x_{i,12} - x_{i,22}| \right)}{\sum_{i=1}^{n} \left(|x_{i,11} - x_{i,21}| + |x_{i,12} - x_{i,22}| \right)^2 / r_i} \tag{15}$$

and for $\Phi_2(P_{i1}, P_{i2})$:

$$k = \frac{\sum_{i=1}^{n} \left(|x_{i,11} - x_{i,21}|^2 + |x_{i,12} - x_{i,22}|^2 \right)^{1/2}}{\sum_{i=1}^{n} \left(|x_{i,11} - x_{i,21}|^2 + |x_{i,12} - x_{i,22}|^2 \right) / r_i} \tag{16}$$

We would like the reader to recall that $\pi_i = \langle P_{i1}, P_{i2} \rangle$ is the ith **pair** of points.

However, the calculations for k and p of $\Phi_3(P_{i1}, P_{i2})$, and k, p and s of $\Phi_4(P_{i1}, P_{i2})$ are slightly more complicated. They require the solution of the following unconstrained optimization problems:

$$\min_{k,p} \sum_{i=1}^{n} \left[\frac{k \left(|x_{i,11} - x_{i,21}|^p + |x_{i,12} - x_{i,22}|^p \right)^{1/p} - r_i}{\sqrt{r_i}} \right]^2 \tag{17}$$

$$\min_{k,p,s} \sum_{i=1}^{n} \left[\frac{k \left(|x_{i,11} - x_{i,21}|^p + |x_{i,12} - x_{i,22}|^p \right)^{1/s} - r_i}{\sqrt{r_i}} \right]^2 \tag{18}$$

Although they are quite simple with respect to the number of variables (which is two and three), the number of nonlinearities introduced by these problems can be very large depending on the size of the pairs of points within the training set. Their minimizations can be carried out by using any known non-linear optimization package, such as MINOS 5.1 [40]. To render the computations faster, we wrote our own optimization procedures rather than invoke MINOS 5.1. The results using distance functions are given in Table 2. Observe that the rectilinear distance metric has the worst performance. Meanwhile the accuracy of $\Phi_4(P_a, P_b)$ is the highest. These facts definitely support our previous inference on the arbitrariness of the road structure in Türkiye.

Table 2: Average error on the test set per pair using the four parametric distance functions.

ESTIMATOR	PARAMETERS	TRAINING ERROR	TEST ERROR
$\Phi_1(P_a, P_b)$	k=1.082	8.024	12.396
$\Phi_2(P_a, P_b)$	k=1.320	3.650	8.410
$\Phi_3(P_a, P_b)$	k=1.286 p=1.688	3.360	8.940
$\Phi_4(P_a, P_b)$	k=1.286 p=1.688 s=1.829	3.360	8.940

Table 3: Average error of the neural network and combining estimators.

ESTIMATOR	TRAINING ERROR	TEST ERROR
Multi-layer perceptron	2.63	7.91
Regression neural network	2.38	8.49
Voting	2.26	7.63
Stacking	3.81	7.41

4.3 Perceptron Based Methods

In their recent study Alpaydın *et. al.* [2] employed a multi-layer perceptron with one hidden layer with the back-propagation learning rule. In this work the best input representation, namely the vector **u** was determined as the four data values, and the Euclidean distance in between was supplemented as a hint:

$$\boldsymbol{u} = (x_{a1}, x_{a2}, x_{b1}, x_{b2}, \|P_a - P_b\|)^T$$

In terms of output representation, the authors found out that learning the ratio of actual distance to Euclidean distance, namely the directional bias [11, 13], is better than learning the actual distance itself :

$$y = r/\|P_a - P_b\|$$

Here y simply denotes the output of the perceptron.

This can be perceived as a case of extending $\Phi_2(P_a, P_b)$ in the parametric case. Instead of computing a single global constant k, it is as if the neural network computes a continuous function $k(P_a, P_b)$ by which it scales the Euclidean distance. Note that in these two cases, they can take advantage of the *a priori* knowledge that $\Phi(P_a, P_b) = \Phi(P_b, P_a)$ and effectively double the training set. Details on the implementation, and results with regression neural networks [56] and combining estimators, which use voting [1] and stacking [9, 61] strategies can be found in the same work. We summarize these results in Table 3. The training data they have used is a subset of ours, although the test set is the same.

4.4 Discrete Vector Quantization and The Self-Organizing Map

The very first step involved in implementing the discrete VQ-algorithm was to build a grid structure with a specific cell length, which actually indirectly specifies the resolution. One should note that as the cell length decreases the resolution increases resulting in a finer grid structure and vice versa. In our experimental setting this was achieved as follows.

In all the experiments reported in earlier publications involving Türkiye [4, 5, 46] , the original map of Türkiye was enclosed within a bounding rectangle defined between the latitudes and longitudes 36^o N, 26^o E, 42^o N and 45^o E respectively. The coordinates of the cities in the database were obtained by plotting the national boundary, the cities and towns, and the grid on a large map in which Ankara, the capital, was placed as the origin. Thus the coordinate axes for the bounding rectangle had the equations $X = -545.9$, $Y = -430.1$, $X = 1013.2$, and $Y = 243.1$ respectively. Consequently, the data points representing the cities had coordinate values within these ranges, and thus Mardin had coordinates $(666.4, -244.8)$. Similarly, Sorgun had the coordinates $(193.8, -5.1)$ and the actual road distance between Mardin and Sorgun was known to be 703.5 kms. Since the data points *do not* all lie on a grid discretizing must be effected in the process of the computation. This is quite easily done by multiplying every coordinate by a factor, called the *magnifying factor*, specified by μ. The new coordinates are then rounded off to the closest integer. Note that by multiplying the coordinates by μ and rounding, the integers are mapped equivalently to the model proposed earlier except for a simple translation. Indeed, in any actual implementation, this translation need not be done, since the computations are not effected whether we work in the range $[0, .., J]$ or in the range $[-a, ..., b]$. Observe that, what we lose (from going from the continuous world to the discretized) is the fractional portions of the coordinates, but what we gain is that these fractional portions are magnified by μ and then rounded off to the closest integer. Thus when the resolution parameter μ is 4, Mardin and Sorgun have the *discretized* coordinates $(2666, -979)$ and $(775, -20)$ respectively. We however emphasize that although the coordinates of the points are discretized, and thus have their magnitudes magnified, the *actual* distances between the corresponding cities are still the real world travel distances, and are thus unchanged.

To demonstrate the power of our strategy we have done *numerous* experiments involving initial random and non-random partitions for Türkiye. In the interest of brevity we shall merely report some of these results which highlight the characteristic features of the scheme. The rest of the results can be found in the doctoral thesis of N. Aras, which is currently being prepared. Also, in the interst of comparing the current discretized work with its continuous counterpart [5, 46], the results which we report and the initial partitions are exactly the same as those for which we had reported earlier results in [5, 46].

In each of the figures, the towns themselves are marked with an '$\times$'. Note that the actual map of Türkiye has not been superimposed in the figure to avoid cluttering it. The twelve squares, '$\Box$' represent the final positions of the code-book vectors.

Consider Figure 1. In this case the initial partitioning had four sub-regions and was achieved "manually" but in an arbitrary manner. The sub-regions divided the country into four rectangles and shown in the figure, each containing 20 towns in the training set L. To demonstrate the power of the strategy, we used a DVQ strategy with **only** 3 code-book vectors in each region which were initialized to be on the border of their representative regions. During the intra-regional polarizing phase the DVQ algorithm was invoked with a value of α which started at unity and decreased linearly to 0.9 in 1,000 iterations. As expected, most of the learning was accomplished in this phase. Thereafter, in the fine tuning phase the value of α was drastically switched to 0.2 and decreased linearly to attain to 0.1 in 2,000 time steps. In the inter-regional polarizing phase, the value of α was maintained to 0.1. The constant for the migration of code-book vectors from the same class, ϵ was maintained at 0.25 and done for 2000 iterations (i.e., 25 cycles of all the 80 training sites). Indeed, the entire convergence for both these phases took only a matter of a couple of seconds.

The DVQ was implemented for various values of the magnifying factor μ ranging from 1 to 32 in powers of 2. Thus, the coarsest resolution with $\mu = 1$ corresponded to the case when the original data was rounded off, and as explained earlier, larger values of μ resulted in data points which were rounded off after multiplying the co-ordinates by μ. It was generally observed that for small values of μ, both the intra and inter-regional polarizing effected the code-book vectors. But as the value of μ increased (typically more than 4) most of the polarizing was effected by the intra-regional polarizing and the inter-regional polarizing merely verified the locations of the final code-book vectors without invoking any additional changes. In each case, the final partitioning (after the code-book vectors converged) was fully determined by the discriminant function implicitly created by the bisectors of the lines joining the code-book vectors. This partitioning is adaptively learnt and is shown in Figure 1 in bold lines for the case when $\mu = 8$. Observe the power of the adaptive regional partitioning scheme.

After the intra and inter-regional learning, the constants for the underlying patchwork functions

Table 4: A comparison of the average error for the single functional form for each subregion where the continuous and discretized VQ schemes are used.

ESTIMATOR	CONTINUOUS TRAINING ERROR	CONTINUOUS TEST ERROR	DISCRETE TRAINING ERROR($\mu = 8$)	DISCRETE TEST ERROR($\mu = 8$)
Simple Average	3.34	7.83	3.363	7.836
$\Phi_2(P_a, P_b)$	3.12	8.05	3.141	8.046
$\Phi_3(P_a, P_b)$	2.93	7.80	2.926	7.731
$\Phi_4(P_a, P_b)$	2.77	7.60	2.766	7.523

Table 5: A comparison of the average error for a separate sub-function for each code-book vector where the continuous and discretized VQ schemes are used.

ESTIMATOR	CONTINUOUS TRAINING ERROR	CONTINUOUS TEST ERROR	DISCRETE TRAINING ERROR($\mu = 8$)	DISCRETE TEST ERROR($\mu = 8$)
Simple Average	2.42	7.69	2.433	7.622
$\Phi_2(P_a, P_b)$	2.36	7.90	2.371	7.793
$\Phi_3(P_a, P_b)$	2.11	7.48	2.091	7.325
$\Phi_4(P_a, P_b)$	1.93	7.12	1.917	7.077

were estimated using the *true coordinates* of the training sites and their corresponding recorded distances. The estimates we computed were of two sorts. First of all, to show the power of the multi-regional approach, we assumed that the distances within each region and the distances between the regions were each characterized by a single functional form. Thus, since we partitioned G into four sets, a functional form of the type (13) involved 10 constants. These constants were estimated by both a simple averaging scheme and a VQ method as explained in Section 3.7. When the explicit form of each intra and inter-regional function was of the types $\Phi_3(P_a, P_b)$ and $\Phi_4(P_a, P_b)$ (where the parameters to be estimated were $\{k, p\}$ and $\{k, p, s\}$), the total number of parameters to be estimated was 20 and 30 respectively. In the latter two cases, the optimization was done independently, and this was typically more time consuming because it involved invoking separate nonlinear optimization procedures. The results which we have obtained are quite remarkable and are tabulated in Table 4, where the average training and testing errors are recorded for the case when $\mu = 8$. For example, when the functional form was assumed to be of type $\Phi_2(P_a, P_b)$, the average error obtained by averaging k (which was exactly the error obtained by a VQ algorithm in the k-space) was 7.836. This decreased to 7.731 and 7.523 for the cases when the parameters were $\{k, p\}$ and $\{k, p, s\}$ respectively. The corresponding results for averaging in the k-space using the continuous VQ solution was 7.83, which decreased to 7.80 and 7.60 for the cases when the parameters were $\{k, p\}$ and $\{k, p, s\}$ respectively. The results for both the algorithms are tabulated in Table 4.

The full power of the multi-regional approach is clearly displayed if we "patch" the distance function using a separate sub-function **for** each code-book vector and **between** each code-book vector. In this case, since we partitioned G into four sets with 3 code-book vectors in each region, we would involve 78 explicit sub-functions. A functional form of the type (13) would now involve 78 constants, and when the forms of each intra and inter-regional function were of the types $\Phi_3(P_a, P_b)$ and $\Phi_4(P_a, P_b)$ (where the parameters to be estimated were $\{k, p\}$ and $\{k, p, s\}$, the total number of parameters to be estimated was 156 and 234 respectively. Again, as in the above, the latter two cases involved an independent non-linear optimization. The results which we have obtained are truly amazing, and are given in Table 5.

In the most conservative case, the testing error is only 7.622, and in the case when the functions are

characterized by $\{k, p, s\}$ the test error went as low as 7.077. This should be compared with the results for the continuous VQ scheme [5, 46] where the most conservative case (obtained by averaging in the k-space) yielded a testing error of 7.69, and in the case when the functions are characterized by $\{k, p, s\}$ the test error was 7.12. The power of the scheme is clear - it was generally observed that for this value of μ, the magnifying parameter, the discretized scheme performed uniformly better than the continuous scheme which, to our knowledge, has been the most accurate scheme reported in the literature.

Observe too that like the continuous scheme [5, 46] the most time consuming phase of the learning is the optimization stage. But since this is done only once (during the training phase) the work done is well worth its while. But unlike the continuous scheme, all the learning is performed using only simple integer computations without even evaluating the Eucledian norm between points. Thus, from the point of view of both speed and accuracy, our current scheme seems to be the most superior scheme currently available. Subsequently, in the testing phase, the estimation of the distance between any two points merely involves computing the Discretized Euclidean distance between them and invoking the computation of the functional form associated with their nearest code-book vectors. We are currently investigating how the optimization phase (in $\{k\}$, $\{k, p\}$ or $\{k, p, s\}$) can itself be circumvented by using a VQ algorithm in the corresponding parameter space. This will, of course, involve a gradient descent type of algorithm for each node-pair and distance processed. But since the criterion functions are highly nonlinear deriving such a gradient method is not trivial.

A word regarding the variation of the accuracy with the magnifying parameter is not out of place. Generally speaking, the accuracy is comparable to the other reported schemes (other than the continous VQ scheme [5, 46]) for small values of μ. This accuracy increases remarkably with the magnification as μ increases from 2 to 8 and then tends to stabilize thereafter. This implies that we can use the effect of magnification and discretization profitably only till a certain limit. Beyond this limit, magnifgying and rounding yields no (incremental) marginal advantage. The variation of the test error as a function of μ (drawn on a logarithmic scale) is shown in Figure 2 for the case when the functions are characterized by $\{k, p, s\}$. Notice that this error starts at the value of 7.907 when $\mu = 1$, and decreases to the value of 7.526 for $\mu = 2$. It further decreases to the value of 7.077 for $\mu = 4$, stays at this value till $\mu = 256$. We have observed that this performance is typical.

From Figure 1, we see that the final regional boundaries do not differ "significantly" from the original "arbitrary" ones. The difference between the two sets of boundaries would have been a lot more accentuated if the initial boundaries had been more randomly generated. To demonstrate this we now report the results for a case when the initial quadrilaterals are randomly generated. To achieve this we divided the bounding rectangle of Türkiye into four "random" quadrilaterals. This was achieved by generating a random point on each of the four edges of the bounding rectangle. The lines joining the points on the opposite sides of the bounding rectangle were now used to constitute the four quadrilaterals.

As in the previous case the initial random partitions are shown in Figure 3. In each sub-region the number of code-book vectors was 3. A DVQ strategy with $\mu = 8$ with **only** these code-book vectors in each region was invoked with values of α, ϵ and the number of cycles being as in the above experiment. As in Figure 1, the final partitioning (after the code-book vectors converged) was fully determined by the bisector discriminant functions and is also shown in Figure 3 in bold lines. The power of the method is clear because even though we have used a random partitioning, the final partitioning yields reasonably good results. Indeed, after the intra and inter-regional learning, the constants for the underlying patchwork functions were estimated using the training sites and their corresponding recorded distances as in the above case. In the interest of brevity we merely report the error when the the explicit form of each intra and inter-regional function was of the type $\Phi_4(P_a, P_b)$ when we "patched" the distance function using a separate sub-function **for** each code-book vector and **between** each code-book vector. As in the previous case these characterizing constants were computed by an optimization procedure. In this case the training error was 1.856 and the testing error was as low as 6.987 which is far superior to all the previously reported methods. Note that in the random case cited for the continuous VQ algorithm [5, 46], the corresponding errors were 1.787 and 7.189 respectively.

Although we are aware of the fact that the initial boundaries influence the final ones, the way by which they influence them is still unknown in both the continuous and discretized scenarios. In general, although these results for the random case are so promising, we believe that it is disadvantageous to partition the cities in a completely random way, because it would defeat the very purpose of partitioning - which

attempts to take advantage of the geographical proximity between cities in the various sub-partitions. However, in any practical setting it is sufficient that we find *some* initial partitions using which excellent classification and testing accuracy are obtainable. In our case, from the above results we can see that we are able to get results which are the best results obtainable from any single or hybrid scheme. Unlike for for the continuous VQ algorithm [5, 46] we have not been able to determine any initial 2-partitions with 6 code-book vectors in each which can yield superior classification and testing. We are currently investigating how we can explicitly use "intelligent" information (georgraphical proximity information) in the initial partitioning to yield even superior results.

5 CONCLUSIONS AND DISCUSSIONS

In this paper we have studied the problem of estimating arbitrary distance functions. To achieve this we utilized the learning concepts involving two vastly different areas of adaptive learning namely, neural networks and learning automata. Indeed, we have developed a method by which the general philosophies of Vector Quantization (VQ) and discretized automata learning can be incorporated to yield Discretized Vector Quantization (VQ). We have also studied the estimation problem in its generality - the assumptions made on the arbitrary distance function, Φ, are quite relaxed : The set of inter-node distances dictated by Φ may or may not satisfy all the rigorous properties of a well-defined mathematical norm. Furthermore, the triangular inequality may also be violated. However, to keep the informal concepts of a distance measure valid, we impose the requirement that Φ is loosely related to the Euclidean norm, and so if P_i, P_j, P_m and P_n be any four nodes, if the pairs (P_i, P_j) and (P_m, P_n) are "close" to each other, the respective arbitrary distances between (P_i, P_m) and (P_j, P_n) must be correspondingly of *similar* magnitude. Also, we assume that the explicit form of this distance function is both unknown and uncomputable.

Unlike traditional Operations Research methods, which use parametric distance functions, we have utilized DVQ principles to first adaptively polarize the nodes into sub-regions. Subsequently, the parameters characterizing the sub-regions are themselves learnt using by a variety of methods including a distinct VQ strategy in the (meta) parameter-domain.

The algorithms have been rigorously tested for the actual road-travel distances involving cities and towns in Türkiye. They converge very quickly – in a matter of seconds, and the numerical results obtained are conclusive. Indeed, they are among the best results currently available from any **single or hybrid** strategy and are often superior even to the case when continuous VQ was used for the polarizing [5, 46]. The results of Alpaydın *et. al.* [2] show how a combination of learning strategies can be used to yield superior results by incorporating stacking and voting principles. Clearly, such principles can be used subsequent to our current results to yield even smaller estimation errors.

The salient feature of our present work is that it is, to the best of our knowledge, the pioneering paper which merges the fields of learning automata and neural networks to yield a discretized"adaptive" multi-regional approach to distance estimation. The regions are learnt adaptively using discriminant functions derived implicitly from the code-book vectors. In this process the algorithms perform simple integer manipulations and are thus extremely fast. Subsequent to the partitioning, the actual parameters of the intra-regional and inter-regional functions can be obtained either by optimization (in a non-all-neural approach, for example when k, p and s are parameters) or by using a VQ algorithm in this parameter space itself. This is also novel, because the problem lends itself to many distinct philosophies of learning. Finally, arguing as in [5, 46], we believe that the VQ and its discretized counterpart are superior to the Perceptron based methods because unlike the latter, the distance function Φ itself is defined on a well-defined Euclidean space. Consequently, learning the weights (the code-book vectors) within this space is a much more natural characterization than learning it in a space where the weight vectors have no physical significance.

Acknowledgments— John Oommen is partially supported by the National Sciences and Engineering Research Council (NSERC) of Canada and by a travel grant from (TÜBİTAK) - BAYG foreign scientist support program. İ. Kuban Altınel and Necati Aras are partially supported by the Turkish Scientific and Technical Research Council (TÜBİTAK) Grant TBAG-1336 and the Boğaziçi Research Found Grant 96A0361.

References

[1] E. ALPAYDIN, 1993. Multiple Networks for Function Learning, *IEEE International Neural Network Conference*, San Francisco, vol 1, March, 9–14.

[2] E. ALPAYDIN, İ. K. ALTINEL and N. ARAS, 1996. Parametric Distance Functions vs. Nonparametric Neural Networks for Estimating Road Travel Distances, *European Journal of Operational Research* (to appear).

[3] İ. K. ALTINEL and N. ARAS, 1994. Estimating Road Distances in İstanbul with Single and Multi Regional Models, Research Paper Series No : FBE-IE- 05/94-05, Department of Industrial Engineering, Boğaziçi University,İstanbul.

[4] İ. K. ALTINEL, N. ARAS, A. ALIE, G. CANGÜR, R. ÖZEL and A. YÜCEL, 1994. Estimating Road Travel Distances in Türkiye, Research Paper Series No : FBE-IE-03/94-03, Department of Industrial Engineering, Boğaziçi University, İstanbul.

[5] İ. K. Altınel, J. Oommen & N. Aras. (1995) Vector Quantization for Arbitrary Distance Function Estimation. *ORSA Journal on Computing* (Being Revised).

[6] W. BERENS, 1988. The Suitability of the Weighted L_p norm in Estimating Actual Road Distances, *European Journal of Operational Research 34*, 39–43.

[7] W. BERENS and F. KÖRLING, 1985. Estimating Road Distances by Mathematical Functions, *European Journal of Operational Research 21*, 54–56.

[8] W. BERENS and F. KÖRLING, 1988. On Estimating Road Distances by Mathematical Functions–A Rejoinder, *European Journal of Operational Research 36*, 254–255.

[9] L. BREIMAN, 1992. Stacked Regression, TR-367, Department of Statistics, University of California, Berkeley.

[10] J. BRIMBERG, P. D. DOWLING and R. F. LOVE, 1994. The Weighted One-two Norm Distance Model : Empirical Validation and Confidence Interval Estimation, *Location Science 2*, 91–100.

[11] J. BRIMBERG and R. F. LOVE, 1992. A New Distance Function for Modeling Travel Distances in a Transportation Network, *Transportation Science 26*, 129–137.

[12] J. BRIMBERG, R. F. LOVE and J. H. WALKER, 1995. The Effect of Axis Rotation on Distance Estimation, *European Journal of Operational Research 80*, 357–364.

[13] J. BRIMBERG and G. O. WESOLOWSKY, 1992. Probabilistic L_p Distances in Location Models, *Annals of Operations Research 40*, 67–75.

[14] R. O. DUDA and P. E. HART, 1973. *Pattern Classification and Scene Analysis*, John Wiley.

[15] H. ERKUT and S. POLAT, 1992. A Simulation Model for a Urban Fire Fighting System, *Omega 20*, 535–542.

[16] R. A. FILDES and J. B. WESTWOOD, 1978. The Development of Linear Distance Functions for Distribution Analysis, *Journal of the Operational Research Society 29*, 585–592.

[17] R. L. FRANCIS, L. F. MCGINNIS Jr. and J. A. WHITE, 1992. *Facility Layout and Location : An Analytical Approach, 2nd edition*, Prentice Hall, Englewood Cliffs.

[18] K. FUKUNAGA , *Introduction to Statistical Pattern Recognition, 2nd edition*, Academic Press, San Diego.

[19] D. H. GRAF and W. R. LALONDE, 1988. A Neural Controller for Collision-free Movement of General Robot Manipulators, *Proc. IEEE Int. Conf. on Neural Networks*, I77–I-84.

[20] D. H. GRAF and W. R. LALONDE, 1989. Neuroplanners for Hand/Eye Coordination, *Proc. Int. Joint Conf. on Neural Networks*, II-543–II- 548.

[21] R. M. GRAY, 1984. Vector Quantization, *IEEE ASSP Mag.*, vol 1, 4–29.

[22] A. HEMANI and A. POSTULA, 1990. Scheduling by Self Organisation, *Proc. Int. Joint Conf. on Neural Networks, IJCNN-90-WASH-DC*, II-543-II-546.

[23] T. KOHONEN, 1988. The "Neural" Phonetic Typewriter, *Computer 21*, 11–22.

[24] T. KOHONEN, 1990. The Self-Organizing Map, *Proc. IEEE*, vol 78, 1464–1480.

[25] T. KOHONEN, 1995. *Self-Organizing Maps*, Berlin, Heidelberg, Germany : Springer - Verlag.

[26] T. KOHONEN, K. MAKISARA and T. SARAMAKI, 1984. Phonotopic Maps – Insightful Representation of Phonological Features for Speech Recognition, *Proc. Seventh. Int. Conf. on Pattern Recognition*, 182–185.

[27] T. KOHONEN, K. TORKKOLA, M. SHOZOKAI, J. KANGAS and O. VENTA, 1987. Microprocessor Implementation of a Large Vocabulary Speech Recognizer and Phonetic Typewriter for Finnish and Japanese, *Proc. European Conference of Speech Technology*, 377– 380.

[28] S. LAKSHMIVARAHAN, 1981. *Learning Algorithms : Theory and Applications*, New York : Springer - Verlag.

[29] Y. LINDE, A. BUZO and R. M. GRAY, 1980. An Algorithm for Vector Quantization," *IEEE Trans. Communication COM-28*, 84–95.

[30] R. F. LOVE and J. G. MORRIS, 1972. Modeling Inter-City Road Distances by Mathematical Functions, *Operational Research Quarterly 23*, 61–71.

[31] R. F. LOVE and J. G. MORRIS, 1979. Mathematical Models of Road Travel Distances, *Management Sciences 25*, 130–139.

[32] R. F. LOVE and J. G. MORRIS, 1988. On Estimating Road Distances by Mathematical Functions, *European Journal of Operational Research 36*, 251–253.

[33] R. F. LOVE, J. G. MORRIS and J. WESOLOWSKY, 1988. *Facilities Location : Models and Methods*, North - Holland, New York.

[34] R. F. LOVE and J. H. WALKER, 1994. An Empirical Comparison of Block and Round Norms for Modeling Actual Distances, *Location Science 2*, 21–43.

[35] R. F. LOVE, J. H. WALKER and M. L. TIKU, 1995. Confidence Intervals for $l_{k,p,\theta}$ Distances, *Transportation Science 29*, 93–100.

[36] J. MAKHOUL, S. ROUCOS and H. GISH, 1985. Vector Quantization in Speech Coding, *Proc. IEEE*, vol. 73, 1551–1588.

[37] K. M. MARKS and K. F. GOSER, 1988. Analysis of VLSI Process Data Based on Self-Organizing Feature Maps, *Proc. Neuro-Nimes'88*, 337–347.

[38] J. MARTINETZ, H. J. RITTER and K. J. SCHULTEN, 1990. Three-dimensional Neural Net for Learning Visuomotor Coordination of a Robot Arm, *IEEE Trans. Neural Networks*, 131–136.

[39] A. K. MITTAL and V. PALSULE, 1984. Facilities Location with Ring Radial Distances, *Institute of Industrial Engineers Transactions 16*, 59–64.

[40] B. A. MURTAGH and M. A. SAUNDERS, 1983 (revised 1987). MINOS 5.1 User's Guide, Technical Report No : SOL 83 - 20R, Stanford University, Stanford, California.

[41] K. S. NARENDRA and M. A. L. THATHACHAR 1989. *Learning Automaton : An Introduction*, Englewood Cliffs, New Jersey, Prentice - Hall.

[42] E. K. NEUMANN, D. A. WHEELER, A. S. BURNSIDE, A. S. BERNSTEIN and J. C. HALL, 1990. A Technique for the Classification and Analysis of Insect Courtship Song, *Proc. Int. Joint Conf. on Neural Networks, IJCNN-90-WASH-DC*, II-257-II-262.

[43] B. J. OOMMEN 1986. Absorbing and Ergodic Discretized Two Action Learning Automata, *IEEE Transactions on Systems, Man and Cybernetics SMC-16*, 282–293.

[44] B. J. OOMMEN and J. K. LANCTÔT 1990. Discretized Pursuit Learning Automata, *IEEE Transactions on Systems, Man and Cybernetics SMC-20* 431–438.

[45] B. J. OOMMEN, N. ANDRADE and S. S. IYENGAR 1991. Trajectory Planning of Robot Manipulators in Noisy Workspaces Using Stochastic Automata, *International Journal of Robotics Research April 1991* 135–148.

[46] B. J. OOMMEN, İ. K. ALTINEL and N. ARAS 1995. Arbitrary Distance Function Estimation Using Vector Quantization*Proc. IEEE International Conference on Neural Networks 6* 3062–3067.

[47] J. K. LANCTÔT and B. J. OOMMEN 1992. Discretized Estimator Learning Automata, *IEEE Transactions on Systems, Man and Cybernetics SMC-22* 1473–1483.

[48] B. J. OOMMEN and D. C.Y. MA, 1992. Stochastic Automata Solutions to the Object Partitioning Problem, *The Computer Journal 35* A105–A120.

[49] G. A. ORLANDO, R. MANN and S. HAYKIN, 1990. Radar Classification of Sea-ice Using Traditional and Neural Classifiers, *Proc. Int. Joint Conf. on Neural Networks, IJCNN-90-WASH-DC*, II-263-II-266.

[50] J. PERREUR and J. THISSE, 1974. Central Metrics and Optimal Location, *Journal of Regional Science 14*, 411–421.

[51] J. POTVIN, 1993. The Travelling Salesman Problem: A Neural Network Perspective, *ORSA Journal on Computing 5*, 328–348.

[52] H. J. RITTER, J. MARTINETZ and K. J. SCHULTEN, 1989. Topology Conserving Maps Learning Visuomotor Coordination, *Neural Network 2*, 159–168.

[53] H. J. RITTER and K. J. SCHULTEN, 1986. Topology Conserving Mappings for Learning Motor Tasks, *Proc. Neural Networks for Computing, AIP Conference*, 376– 380.

[54] H. J. RITTER and K. J. SCHULTEN, 1988. Extending Kohonen's Self-organizing Mapping Algorithm to Learn Ballistic Movements, *NATO ASI Series*, 393– 406.

[55] J. K. SAMARABANDU and O. E. JAKUBOWICZ, 1990. Principles of Sequential Feature Maps in Multi-Level Problems, *Proc. Int. Joint Conf. on Neural Networks, IJCNN-90-WASH-DC* II-683-II-686.

[56] D. F. SPECHT, 1991. A General Regression Neural Network, *IEEE Transactions on Neural Networks 2*, 568–576.

[57] V. TRYBA, K. M. MARKS, U. RÜTCKER and K. GOSER, 1988. Selbst-organizierende Karten Als Lernende Klassifizierende Speicher, *IFG Fachbericht 102*, 407–419.

[58] M. L. TSETLIN, 1973. *Automaton Theory and the Modelling of Biological Systems*, New - York, Academic Press.

[59] J. E. WARD and R. E. WENDELL, 1980. A New Norm Measuring Distance which Yields Linear Location Problems, *Operations Research 28*, 836–844.

[60] J. E. WARD and R. E. WENDELL, 1985. Using Block Norms for Location Modeling, *Operations Research 33*, 1074–1091.

[61] D. H. WOLPERT, 1992. Stacked Generalization, *Neural Networks 5*, 241–259.